VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

# Komponenta výukového serveru TI - Kombinatorické hry

# Component of Teaching Server for Theoretical Computer Science - Combinatorial Games

2019 Tomáš Zvolánek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

# Zadání bakalářské práce

Student: **Tomáš Zvolánek**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Komponenta výukového serveru TI - Kombinatorické hry

Component of Teaching Server for Theoretical Computer Science - Combinatorial Games

Jazyk vypracování: čeština

Zásady pro vypracování:

V rámci diplomových a bakalařských prací vzniká výukový server pro předměty teoretické informatiky. Jedná se o sadu dynamických webových stránek umožňujících studentům pochopení různých typů úloh a problémů tím, že si mohou zadat na stránce libovolné zadání a zobrazí se jim řešení včetně postupu. Cílem této bakalářské práce je vytvořit komponentu pro výuku tzv. odebíracích her probíraných v předmětu Teorie her.

1. Nastudujte si oblast kombinatorických her, především tzv. odebíracích her a her Nim.
2. Vytvořte dynamické webové stránky umožňující uživateli následující:
a) zadat počet kupek pro odebírání a pravidla pro jednotlivé tahy,
b) zobrazit si vítěznou strategii včetně zdůvodnění, jak se dá určit (výpočet P a N pozic, Sprague-Grundy funkce, optimálního tahu v dané pozici),
c) hrát danou hru proti počítači (s nastavitelnou pravděpodobností pro optimální tah) nebo proti druhému hráči.
3. Připravte několik ukázkových typů her tak, aby uživatel nemusel pravidla zadávat a mohl hned hrát nebo si zobrazit výpočet optimální strategie.
4. Použité technologie budou voleny tak, aby byly stránky na straně klienta co nejméně platformně závislé a aby mohly být nasazeny na server s již existujícími výukovými stránkami (většina ostatních komponent je založena na ASP .NET nebo javascriptu).

Seznam doporučené odborné literatury:

[1] Thomas S. Ferguson: Game Theory, available online at http://www.math.ucla.edu/~tom/math167.html
[2] Elwyn R. Berlekamp, John H. Conway, Richard K. Guy: Winning Ways for Your Mathematical Plays, Volume 1, A K Peters/CRC Press, 2 edition, 2001

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Martin Kot, Ph.D.**

Datum zadání: 01.09.2018

Datum odevzdání: 30.04.2019


doc. Ing. Jan Platoš, Ph.D.
*vedoucí katedry*

prof. Ing. Pavel Brandštetter, CSc.
*děkan fakulty*

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. dubna 2019

Vrzalek

.....

**Abstrakt**

Tato bakalářská práce se zabývá vývojem webových stránek, které mají být součástí serveru pro výuku teoretické informatiky. Účelem těchto stránek je poskytnou zájemcům o problematiku kombinatorických her (zejména nestranných odebíracích her) nástroj, který jim pomůže dané téma snadněji pochopit. Aplikace umožňuje uživateli vytvořit a hrát vlastní odebírací hru podle zadané konfigurace a v případě potřeby dokáže zobrazit strategii pro optimální hru. Nabízí také všechnu potřebnou teorii, kterou uživatel musí pro zvládnutí hry znát. V textu práce je navíc dokumentován proces vývoje této aplikace, včetně návrhu, použitých technologií a samotné implementace.

**Klíčová slova**: webové stránky, kombinatorické hry, nestranné odebírací hry, teoretická informatika, nim, součty kombinatorických her

**Abstract**

This bachelor's thesis describes the development of a website that is to be a part of a computer science educational server. The aim of the website is to provide a tool for those who are interested in combinatorial games (impartial takeaway games specifically) as an attempt to aid them with their understanding of the topic. The application makes it possible for the user to create and play a game they can configure themselves, and can display the winning strategy if needed. It provides all the neccessary information that a user needs to know before playing the game. The thesis also documents the development process of the application, including the design, implemented technologies, and the implementation itself.

**Key Words**: website, combinatorial games, impartial takeaway games, theoretical computer science, nim, sums of combinatorial games

# Obsah

# Seznam použitých zkratek a symbolů

| | | |
|---|---|---|
| AJAX | – | Asynchronous JavaScript and XML |
| CSS | – | Cascading Style Sheets |
| FTP | – | File Transfer Protocol |
| HTML | – | Hypertext Markup Language |
| JSON | – | JavaScript Object Notation |
| MVC | – | Model View Controller |
| SG | – | Sprague-Grundy |
| UML | – | Unified Modeling Language |
| XML | – | Extensible Markup Language |

# Seznam obrázků

## Seznam tabulek

# Seznam výpisů zdrojového kódu

# 1 Úvod

V rámci výuky teoretické informatiky je vyvíjen server, jehož komponenty mají za cíl usnadnit výuku a pochopení vybraných témat a typů úloh. Tato bakalářská práce se zabývá tvorbou jedné z komponent, konkrétně webovou stránkou pro studium vybraných typů kombinatorických her, které se probírají v předmětu Teorie her.

Stránka musí studentovi, nebo jinému zájemci o danou problematiku, umožnit nastudovat si potřebnou teorii, dále pak vytvořit si odebírací hru s vlastní konfigurací. Vytvořenou hru bude možné hrát proti umělé inteligenci, ale také proti jinému hráči. Je nutné, aby součásti aplikace byla také možnost zobrazit si vítězný postup a strategii dané hry. Díky tomu, že stránky umožní jednoduše vytvořit a modifikovat vlastní hru a následně ji také hrát, by se měla pro uživatele stát daná problematika zábavnější a lépe uchopitelná.

V minulých letech již byla pro tento účel v rámci diplomových prací jedna webová stránka vytvořena. Vzhledem k jejím nedostatkům a nemožnosti je opravit bylo usouzeno, že vhodným řešením je vytvořit nové stránky, které by veškeré nedostatky odstranily.

Kapitola 2 je věnována teorii kombinatorických her, kde se čtenář naučí hry analyzovat a hrát. V kapitole 3 jsou podrobněji popsány důvody vzniku aplikace, požadavky na aplikaci a také uživatelé. Kapitola 4 poskytuje výčet nejdůležitějších technologií, které byly při tvorbě aplikace použity, a také konkrétní místa jejich použití. V kapitole 5 je rozebrán návrh aplikace z různých pohledů. Kapitola 6 se zabývá implementací projektu. Jsou v ní zahrnuty úryvky zdrojového kódu, dále pak ukázky uživatelského rozhraní aplikace a na závěr informace potřebné k úspěšnému nasazení aplikace na server. Závěrečná kapitola 7 obsahuje shrnutí výsledné práce, nabyté zkušenosti a také možnosti vylepšení do budoucna.

# 2 Kombinatorické hry

Kombinatorické hry můžeme definovat jako hry dvou hráčů, kteří mají o hře v každém okamžiku k dispozici úplné informace a provádějí tahy bez jakéhokoli prvku náhody. Hra se skládá z (obvykle konečné) množiny pozic, přičemž se hráči střídají v tazích, kterými se snaží dosáhnout pro ně výhodné pozice. Po dosažení *konečné pozice* (angl. *terminal position*), je jeden hráč prohlášen za vítěze a druhý za poraženého. Konečná pozice je obvykle taková pozice, ze které neexistují žádné další tahy.

Existují dva hlavní typy pravidel, které popisují podmínky vítězství. Podle *normálního pravidla* (angl. *normal play rule*) vyhrává hráč, který provedl poslední tah hry. Naopak při uplatnění *misère pravidla* (angl. *misère play rule)* takový hráč prohrál.

Kombinatorické hry je dále možné rozdělit do dvou kategorií. *Nestranné hry* (angl. *impartial games*) se vyznačují tím, že oba hráči mají z jakékoli pozice k dispozici stejnou množinu tahů. Naopak v případě *partyzánských her* (angl. *partyzan games*), může být množina tahů z konkrétní pozice pro hráče rozdílná. Tato práce se zabývá pouze oblastí nestranných kombinatorických her, hraných podle normálního pravidla [1].

## 2.1 P-pozice a N-pozice

Jak už bylo řečeno, v průběhu hry se hráči svými tahy snaží dostat do konečné pozice, té však obvykle předchází mnoho jiných pozic. Tyto pozice dělíme na takzvané *P-pozice* a *N-pozice*. P-pozice jsou takové, které jsou optimální pro hráče, který zrovna dokončil svůj tah. N-pozice jsou naopak ty, které jsou vítězné pro hráče, který má aktuálně svůj tah provést [2].

P-pozice a N-pozice jsou dále definovány pomocí těchto vlastností [1]:

1. Všechny konečné pozice jsou P-pozice.
2. Z každé N-pozice existuje alespoň jeden tah do P-pozice.
3. Tahem z jakékoli P-pozice se lze dostat výhradně do jedné z N-pozic.

Znalost těchto pozic a jejich charakteristik je důležitá pro získání optimální strategie dané kombinatorické hry.

## 2.2 Odebírací hry

Jednou z podmnožin kombinatorických her jsou hry odebírací. Mějme hromádku žetonů, ze které hráči žetony střídavě odebírají. Kolik žetonů je hráči dovoleno odebrat určíme stanovením *odebírací množiny* (angl. *substraction set*), kde jsou jednotlivé prvky tvořeny celými čísly a reprezentují počet žetonů, které může hráč odejmout. Pokud hráč odebere poslední žeton, nebo po skončení jeho tahu nelze žádný další legální tah provést, je prohlášen za vítěze.

**Příklad:** Příkladem je hra s hromádkou 10 žetonů a odebírací množinou $O = \{1, 2, 3\}$. Do konečné pozice se hra dostane, pokud jeden z hráčů odebere poslední žeton, protože již nelze provést žádný další tah.

K nalezení P-pozic a N-pozic v této hře můžeme použít následující algoritmus, který využívá jejich vlastností:

1. Označme všechny konečné pozice jako P-pozice.
2. Označme všechny pozice, ze kterých se jedním tahem můžeme dostat do nějaké P-pozice, jako N-pozice.
3. Ty pozice, ze kterých je možné táhnout pouze do N-pozic, označíme jako P-pozice.
4. Pokud ve třetím kroku nebyly objeveny žádné nové P-pozice, algoritmus končí. V opačném případě se vracíme do kroku 2.

Výsledek algoritmu pak vypadá takto:

| Počet žetonů | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Pozice | P | N | N | N | P | N | N | N | P | N | N |

Jelikož je hra v počátečním stavu v N-pozici, má hráč, který je na tahu jako první, možnost odebrat optimální množství žetonů. Odebráním dvou žetonů se dostane do P-pozice, která je prozatím vítězná pro hráče, který tah dokončil, tedy pro něj. Za předpokladu, že první hráč po celý zbytek hry neudělá chybu se již nemůže druhý hráč dostat do výhodné pozice.

## 2.3 Hra Nim

Velice známou odebírací hrou je hra Nim. Hra se skládá z libovolného počtu hromádek, kde v každé hromádce může být rovněž jakýkoli počet žetonů. Hráči se střídají v tazích, při kterých si vyberou jednu z hromádek a odeberou z ní žetony. Ve svém tahu může hráč odebrat žetony pouze z jedné hromádky, ale žetonů může odejmout kolik chce (alespoň 1 žeton). Vyhrává hráč, který vezme poslední žeton.

Hru Nim tedy můžeme klidně hrát pouze s jednou hromádkou. Optimální tah je v tu chvíli odebrat všechny žetony najednou. Pokud hru hrajeme se dvěma hromádkami, je pro hráče na tahu výhodné přenést hru do pozice, kdy obsahují obě hromádky stejný počet žetonů. Druhý hráč pak nemá jinou možnost, než svým tahem jednu z hromádek zmenšit. První hráč může opět velikost hromádek vyrovnat. Tímto způsobem mohou hrát tak dlouho, dokud první hráč nevyrovná hromádky do stavu, ve kterém budou obě prázdné. Pozice $(0, 0)$ je totiž konečná, tudíž se první hráč stal vítězem.

S rostoucím počtem hromádek se hra nevyhnutelně stává komplikovanější a méně přehlednou. Rozhodnout, jestli je např. pozice $(11, 5, 12, 20)$ P-pozice, nebo N-pozice je na první pohled velmi obtížné. Určit typ pozice nám pomáhá tzv. *Nim-Součet* (angl. *Nim-Sum*).

**Definice 1:** Nim-Součet dvou nezáporných, celých čísel, je jejich součet v binárním tvaru, kde ale zároveň nepřenášíme jedničky do vyšších řádů. Tato operace se také nazývá $XOR$, neboli *exkluzivní disjunkce* [1].

Všechna nezáporná, celá čísla, mohou být reprezentována jejich dvojkovým ekvivalentem: $x = x_m 2^m + x_{m-1} 2^{m-1} + ... + x_1 2 + x_0$ pro nějaké $m$, kde $x_i$ je buď 1, nebo 0. Zápis $(x_m x_{m-1} ... x_1 x_0)$ pak vyjadřuje číslo v binární formě. Například $18 = 1{\cdot}16 + 0{\cdot}8 + 0{\cdot}4 + 1{\cdot}2 + 0{\cdot}1 = (10010)_2$. Když jsou čísla v binárním tvaru, sečteme číslice v jednotlivých řádech a každý takový součet vydělíme modulo 2. Pokud čísla nemají stejný počet číslic, menší číslo doplníme zleva potřebným počtem nul.

**Příklad:** Operaci XOR provedeme s čísly 18 a 38. Čísla v binárním tvaru: $18 = 10010_2$ a $38 = 100110_2$. Výpočet $18 \oplus 38 = 58$:

$$\begin{aligned} 010010_2 &= 18 \\ \underline{100110_2} &= 38 \\ 110100_2 &= 58 \end{aligned}$$

Pro úplnost je třeba uvést některé z vlastností exkluzivní disjunkce (Nim-součtu).

1. Nim-součet je komutativní, $x \oplus y = y \oplus x$.
2. Nim-součet je asociativní, $x \oplus (y \oplus z) = (x \oplus y) \oplus z$.
3. Platí, že $0 \oplus x = x$ a $x \oplus x = 0$.
4. Pokud $x \oplus y = x \oplus z$, pak $y = z$.

Americký matematik Charles L. Bouton popsal, jakým způsobem nám nim-součet pomáhá s pochopením hry Nim, následující větou:

**Věta:** Pozice $(x_1, x_2, ..., x_m)$ ve hře Nim je P-pozice, právě tehdy když je nim-součet jejích částí roven nule, tedy $x_1 \oplus x_2 \oplus ... \oplus x_m = 0$ [1].

**Příklad:** Vezměme v potaz pozici (14, 6, 10). Zjistíme její nim-součet:

$$\begin{aligned} 1110_2 &= 14 \\ 0110_2 &= 6 \\ \underline{1010_2} &= 10 \\ 0010_2 &= 2 \end{aligned}$$

Jelikož nim-součet není nulový, jedná se o N-pozici. Existuje několik způsobů, jak dojít do P-pozice, například odebrat dva žetony z první hromádky.

$$
\begin{array}{rcl}
1100_2 & = & 12 \\
0110_2 & = & 6 \\
\underline{1010_2} & = & 10 \\
0000_2 & = & 0
\end{array}
$$

## 2.4 Hry na grafech

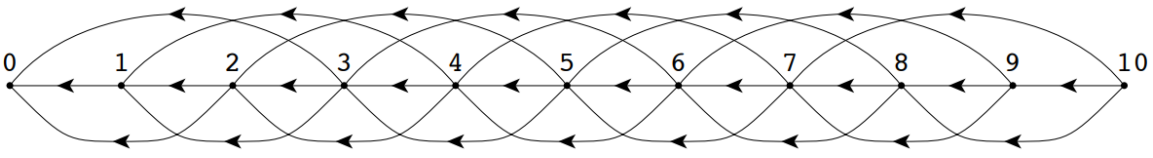
Kombinatorické hry, které jsou popsány v této práci, lze definovat také pomocí orientovaných grafů. Pozice, ve kterých se hra nachází jsou v grafu reprezentovány jeho vrcholy, zatímco hrany grafu reprezentují jednotlivé tahy. Nejprve si uveďme definici orientovaného grafu hry:

**Definice:** Orientovaný graf je dvojice *(X,F)*, kde $X$ představuje neprázdnou množinu vrcholů (pozic) a kde $F$ je funkce, která pro každé $x \in X$ vrací podmnožinu $X$, $F(x) \subset X$. Pro dané $x \in X$ představuje $F(x)$ pozice, do kterých může hráč táhnout z $x$ (tyto pozice jsou následovníky $x$). Pokud je $F(x)$ prázdná, můžeme $x$ nazvat konečnou pozicí [1].

Jako počáteční pozice hry je zvolen jeden z vrcholů, nazveme jej $x_0$. Hráči se pak střídají v tazích, první tah hru přesune do pozice $x_1 \in F(x_0)$. Obecně se hráč každým tahem z konkrétní pozice $x_n$ přesune do nějaké pozice $x_m \in F(x_n)$, jinými slovy do vrcholu, který je následovníkem vrcholu $x_n$. Hráč, který se na začátku tahu ocitne v konečné pozici a nemůže tedy provést vlastní tah, prohrává.

Pod tuto definici však spadají i grafy, na kterých by bylo možné provádět nekonečné mnoho tahů. Pro účely této práce se tedy budeme zabývat pouze grafy, pro které existuje číslo $k$, které je větší nebo rovno délce kterékoli cesty z počátečního vrcholu $x_0$.

**Příklad:** Jako příklad si můžeme znovu uvést odebírací hru s 10 žetony a odebírací množinou $O = \{1,2,3\}$. Každá pozice, včetně té konečné (v tomto případě nastává, pokud je odebrán poslední žeton), je reprezentována jedním z vrcholů grafu $G$, přičemž libovolný vrchol $x_n \in X$, kde $X = \{0, 1, 2, ..., 10\}$. Vrchol $x_0$ představuje prázdnou hromádku žetonů ($x_0 = 0$), nemá tedy žádné následovníky (další žeton již odebrat nelze), proto $F(x_0) = \emptyset$. Dále platí, že $F(1) = \{0\}$, $F(2) = \{0,1\}$ a pro $2 < i \leq 10$, $F(i) = \{k\text{-}3, k\text{-}2, k\text{-}1\}$. Obrázek 1 tuto hru ilustruje.




Obrázek 1: Graf odebírací hry s 10 žetony a $O = \{1,2,3\}$ [1]

## 2.5 Sprague-Grundy funkce

Odebírací hry na orientovaných grafech můžeme popsat pomocí P-pozic a N-pozic. Kromě toho je však můžeme analyzovat *funkcí Sprague-Grundy.*

**Definice:** Funkce Sprague-Grundy grafu (X,F), kde množina $X$ je její definiční obor [1]:

$$g(x) = min\{n \geq 0 : n \neq g(y) : y \in F(x)\}$$

Tato definice říká, že *g(x)* je nejmenší nezáporné celé číslo, které nepatří do množiny hodnot Sprague-Grundy funkce následovníků $x$. Pro ty vrcholy grafu, které reprezentují konečné pozice, platí $g(x) = 0$. Vrchol $x$ totiž nemá žádné následovníky, $F(x) = \emptyset$. Dále platí, že pokud vrchol $x$ následují pouze vrcholy konečné, pak $g(x) = 1$. Stejně tak můžeme rekurzivně najít SG hodnoty pro všechny zbylé vrcholy.

Tímto způsobem můžeme SG funkci použít v případě grafů, které jsme si definovali v kapitole 2.4, nicméně na jiné grafy je třeba použít jiných technik, kterými se tato práce nebude zabývat.

Znalost hodnot SG funkce pro jednotlivé vrcholy grafu nám pomáhá analyzovat konkrétní hru, kterou graf reprezentuje. Pozice $x$, pro které platí $g(x) = 0$, jsou P-pozice. Všechny ostatní jsou N-pozice.

Pro hráče který je na tahu je proto optimální, pokud se svým tahem dostane do pozice $x$, kde $g(x) = 0$. To znamená, že je hra v P-pozici a pokud bude hráč dál provádět optimální tahy, vyhraje.

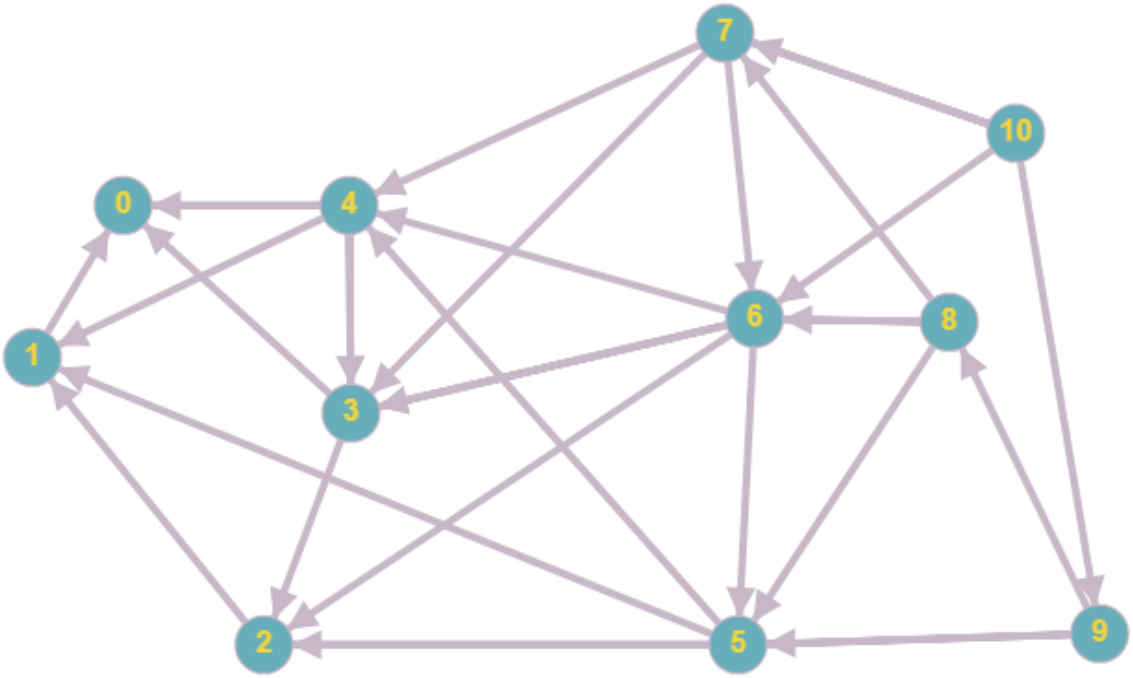
Následující pravidla jsou vlastnosti P-pozic a N-pozic, přepsány do formy, kde využíváme toho, co již víme o hodnotách SG funkce [1].

1. V případě, že je $x$ konečná pozice, pak $g(x) = 0$.
2. Pokud $g(x) = 0$, pak pro všechny pozice $y$, které jsou následovníky $x$, platí: $g(y) \neq 0$.
3. Pokud $g(x) \neq 0$, pak existuje alespoň jedna pozice $y$, která následuje $x$ a zároveň $g(y) = 0$.

**Příklad:** Graf na obrázku 2 opět představuje hru o 10 žetonech, tentokrát ale s odebírací množinou $O = \{1,3,4\}$. Ukážeme si na něm výše zmíněný postup, kterým přidělíme všem jeho vrcholům hodnotu SG funkce. Každý vrchol chápejme jako herní pozici, kde číslo vrcholu představuje počet žetonů, které ve hře zůstaly.

Postup spočívá v tom, že musíme najít vrchol, jehož všichni následovníci už mají hodnotu SG funkce přidělenou. Tomuto vrcholu přiřadíme nejmenší hodnotu, kterou nemůžeme nalézt mezi hodnotami jeho následovníků. Tyto dva kroky budeme opakovat tak dlouho, dokud nebudeme znát SG hodnotu u všech vrcholů grafu.

Nejdříve všem konečným pozicím přidělíme hodnotu 0. Taková pozice je v grafu pouze jedna, a to konkrétně ta, kde zbývá 0 žetonů, tedy vrchol 0. Jediný vrchol, který zná SG hodnoty všech




Obrázek 2: Graf odebírací hry s 10 žetony a $O = \{1,3,4\}$

svých následovníků je vrchol 1. Přidělíme mu hodnotu 1, protože je to nejmenší nezáporné, celé číslo, které nemůžeme u hodnot jeho následovníků nalézt. Následuje vrchol 2. Jeho SG hodnota bude 0. Vrchol 3 bude mít hodnotu 1, vrchol 4 hodnotu 2. Takto budeme pokračovat, dokud nebudou mít všechny vrcholy svoji hodnotu SG funkce:

| Vrchol/počet žetonů | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Hodnota SG | 0 | 1 | 0 | 1 | 2 | 3 | 2 | 0 | 1 | 0 | 1 |

SG funkce poskytuje o hře více informací, než jen jestli je pozice P-pozice, nebo N-pozice. Tyto informace můžeme využít při poněkud komplikovanějších úkonech, jakým je například sčítání kombinatorických her.

## 2.6 Součty kombinatorických her

Z jednotlivých kombinatorických her můžeme tvořit složitější hry podle následujících pravidel. Mějme několik her a každé z nich přidělme počáteční pozici. Hráči se střídají v tazích, při kterých si hráč vybere jednu z her a provede v ní legální tah. Ve zbylých hrách neproběhne žádná změna. Hráči hrají dokud není ve všech hrách dosaženo konečné pozice (další tah není možný). Hráč, který provede poslední tah, vyhrál. Hra, kterou tímto způsobem vytvoříme je *disjunktivní součet* (angl. *disjunctive sum*) jednotlivých her.

Následující věta nám říká, jak můžeme z SG funkcí jednotlivých her získat SG funkci výsledné součtové hry.

**Věta:** Sprague-Grundy funkce součtu kombinatorických her (jejich grafů), je nim-součtem hodnot SG funkce her, ze kterých se skládá [1].

**Příklad:** Vezměme v úvahu dvě triviální odebírací hry, ze kterých složíme hru součtovou. První hra začíná s 10 žetony, druhá se 7. V obou případech bude odebírací množina záviset na aktuálním počtu žetonů dané hry. Hráč může odebrat maximálně polovinu žetonů ze hry, kterou si na začátku tahu vybere.

Tabulka 1: Hra č.1

| Vrchol/počet žetonů | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Hodnota SG | 0 | 0 | 1 | 0 | 2 | 1 | 3 | 0 | 4 | 2 | 5 |

Tabulka 2: Hra č.2

| Vrchol/počet žetonů | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Hodnota SG | 0 | 0 | 1 | 0 | 2 | 1 | 3 | 0 |

Na začátku hry jsou hodnoty SG funkce jednotlivých her $g(10) = 5$ a $g(7) = 0$. Nim-součet je pak $5 \oplus 0 = 5$. První hráč tedy může provést optimální tah odebráním 3 žetonů z první hry: $g(7) \oplus g(7) = 0 \oplus 0 = 0$. Hodnota SG funkce pro součet her je 0 a hráč se tak tímto tahem dostal do P-pozice. Stejně jako tomu bylo v předešlých kapitolách, druhý hráč není momentálně schopen optimálně táhnout. Jakýkoli jeho tah bude do N-pozice a první hráč tak má možnost opět svým tahem přesunout hru do P-pozice. Pokud bude první hráč po zbytek hry dodržovat vítěznou strategii, nemůže prohrát.

# 3 Analýza požadavků

Tato kapitola se bude ve stručnosti zabývat důvodem vzniku webových stránek, požadavky na jejich funkčnost, a také jejich uživateli.

## 3.1 Důvod vzniku aplikace

Stránky mají sloužit jako součást serveru pro podporu výuky teoretické informatiky, konkrétně pak pro jednodušší pochopení některých druhů odebíracích her, dále pak hry Nim a součtových her.

Pro stejný účel už byly v roce 2013 vytvořeny stránky v rámci diplomové práce [9]. Tyto existující stránky však mají několik nevýhod, které by měly nově vznikající stránky odstranit. Mezi hlavní nevýhody patří fakt, že původní stránky nemohou být z důvodu zvolení technologie Java Server Pages nasazeny na stejný server, jako většina ostatních komponent a musí proto být provozovány na samostatném serveru. Další nevýhodou je uživatelské rozhraní, které není dostatečně uzpůsobeno pro mobilní zařízení.

## 3.2 Funkční požadavky

Stránky musí uživateli umožnit vytvořit vlastní odebírací hru, pro kterou může specifikovat pravidla, počet odebíraných žetonů a počet hromádek. Vytvořenou hru bude možno hrát proti druhému hráči, nebo proti umělé inteligenci, která bude disponovat několika obtížnostmi.

Uživatel bude mít při hře možnost zobrazit si vítěznou strategii (včetně optimálních tahů), společně s postupem, jakým se dá určit. Stránky budou obsahovat několik ukázkových typů her, tak aby uživatel mohl ihned hrát.

Technologie budou při implementaci webových stránek vybrány tak, aby mohly být nasazeny na server s již existujícími komponentami. Na straně klienta se očekává nezávislost na používané platformě. Aplikace bude rovněž přizpůsobena mobilním telefonům.

## 3.3 Uživatelé

Aplikace je primárně určena jako učební pomůcka pro studenty teoretické informatiky, ale také pro jejich vyučující. Funkce samotné aplikace nejsou vázány na registraci, nebo jinou formu zadávání osobních údajů, může tedy být využita i veřejností.

# 4 Použité technologie

## 4.1 Microsoft Visual Studio

Visual Studio [8] je integrované vývojové prostředí, které nabízí řadu nástrojů pro vývoj softwaru. Pro tento projekt je toto prostředí vhodné zejména díky zabudované podpoře pro programovací jazyk C# a framework ASP.NET Core MVC, který tvoří základ této bakalářské práce.

## 4.2 ASP.NET Core MVC

ASP.NET Core MVC je multiplatformní framework optimalizovaný pro použití s ASP.NET Core [5]. Podporuje tvorbu dynamických webových stránek a používá nejnovější verzi .NET standardu.

Tento framework podporuje architekturu MVC, která je pro tento projekt použita. Tato architektura odděluje interní logiku aplikace od výstupu, který je renderován uživateli. Rozdělení aplikace na komponenty (model, view, controller) zpřehledňuje její vývoj a usnadňuje údržbu.

## 4.3 Bootstrap

Bootstrap je sada nástrojů pro vývoj pomocí HTML, CSS a JavaScriptu. Tato knihovna je volně ke stažení a je přímo určená k tvorbě responzivních, webových komponent [3].

Bootstrap je také uzpůsoben k využití na různých platformách a zařízeních. Vzhledem k požadavku na využití aplikace i mimo stolní počítače jsou tyto vlastnosti velkou výhodou.

Pro tento projekt byla zvolena šablona Bootswatch Yeti [4].

## 4.4 Newtonsoft Json.NET

Json.NET je knihovna pro práci s daty ve formátu JSON pro platformu .NET. Umožňuje serializovat objekty jazyka C# do formátu JSON a také je deserializovat [6].

## 4.5 jQuery

jQuery je knihovna založená na JavaScriptu, která svým rozhraním zjednodušuje manipulaci s technologiemi HTML, CSS a AJAX. Tato knihovna je otevřený software a nabízí podporu pro širokou škálu webových prohlížečů [7].

## 4.6 ASPone freehosting

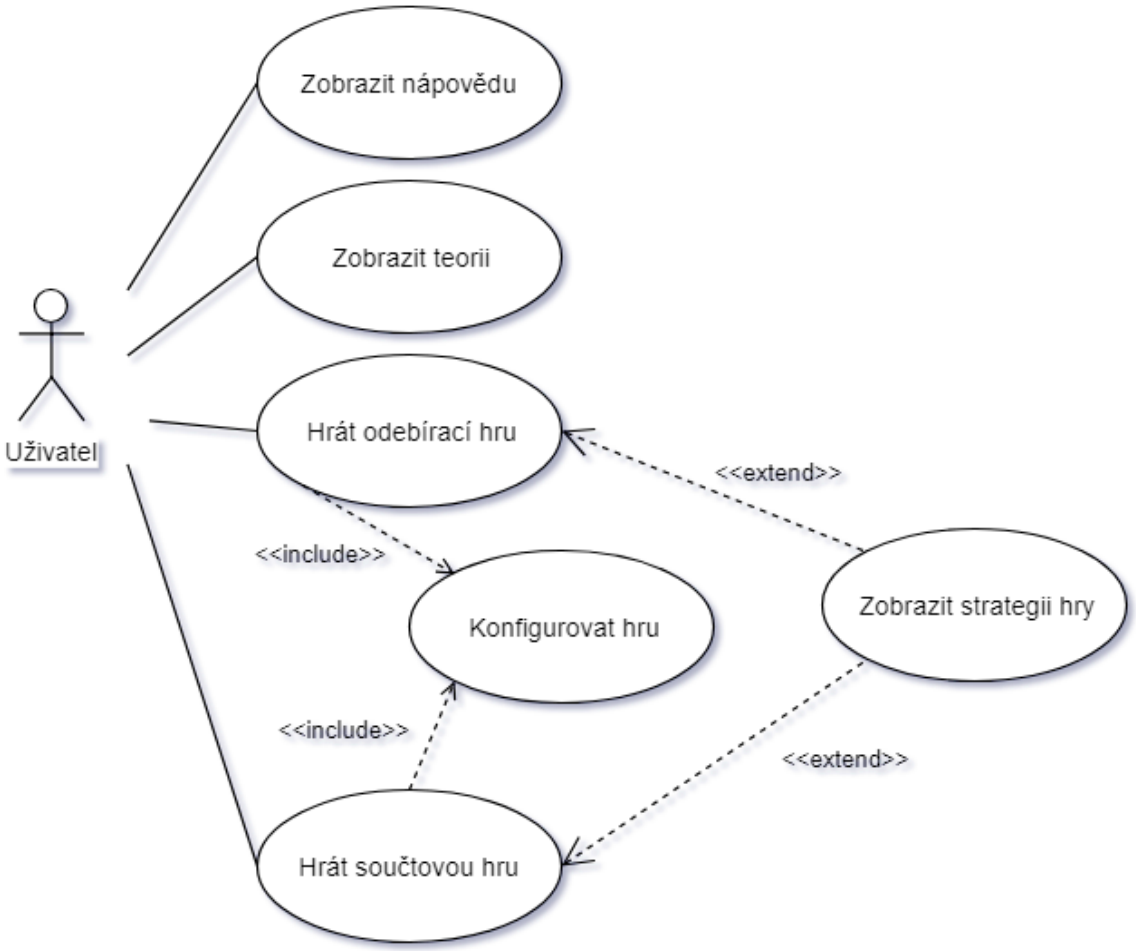
Pro účely testování nasazení aplikace na server a následné testování funkčnosti byl zvolen freehosting *ASPone* [10]. Tato služba podporuje ASP.NET Core MVC a poskytuje dostatečný diskový prostor pro nasazení aplikace.

# 5 Návrh

Tato kapitola se věnuje návrhu aplikace, včetně souvisejících UML diagramů. Popisuje strukturu procesů a komunikaci mezi nimi. Veškeré UML diagramy byly vytvořeny pomocí nástroje *draw.io* [11].

## 5.1 Funkcionalita aplikace

Zde je věnována pozornost právě chování aplikace a různým funkcionalitám, které musí implementovat. Jako první si představíme následující diagram případů užití na obrázku 3.




Obrázek 3: Diagram případů užití

Tento diagram ilustruje jaké chování by měl uživatel od aplikace očekávat, nezaobírá se však tím, jakým způsobem bude toto chování zrealizováno. Nyní si jednotlivé případy užití stručně popíšeme.

1. **Zobrazit nápovědu** - Uživatel bude mít možnost nahlédnout do uživatelské příručky, která bude sloužit jako nápověda u případných nejasností, týkajících se uživatelského rozhraní
2. **Zobrazit teorii** - Aplikace musí poskytovat teorii, která je nezbytná pro pochopení a samotné hraní her.
3. **Hrát odebírací hru** - Nejdůležitější chování aplikace je právě možnost si odebírací hry zahrát.
4. **Hrát součtovou hru** - Uživatel si bude moct zahrát také součtovou hru.
5. **Konfigurovat hru** - Předtím, než si uživatel může zahrát hru, ji musí nakonfigurovat. Konfigurace se ovšem dá přeskočit zvolením náhodných her, v takovém případě hru náhodně konfiguruje server.
6. **Zobrazit strategii hry** - Uživatel bude mít možnost si při hraní projít optimální tahy a strategii. Tento případ rozšiřuje chování aplikace při hraní hry. Hru může uživatel samozřejmě hrát bez toho, aniž by se na strategii podíval.

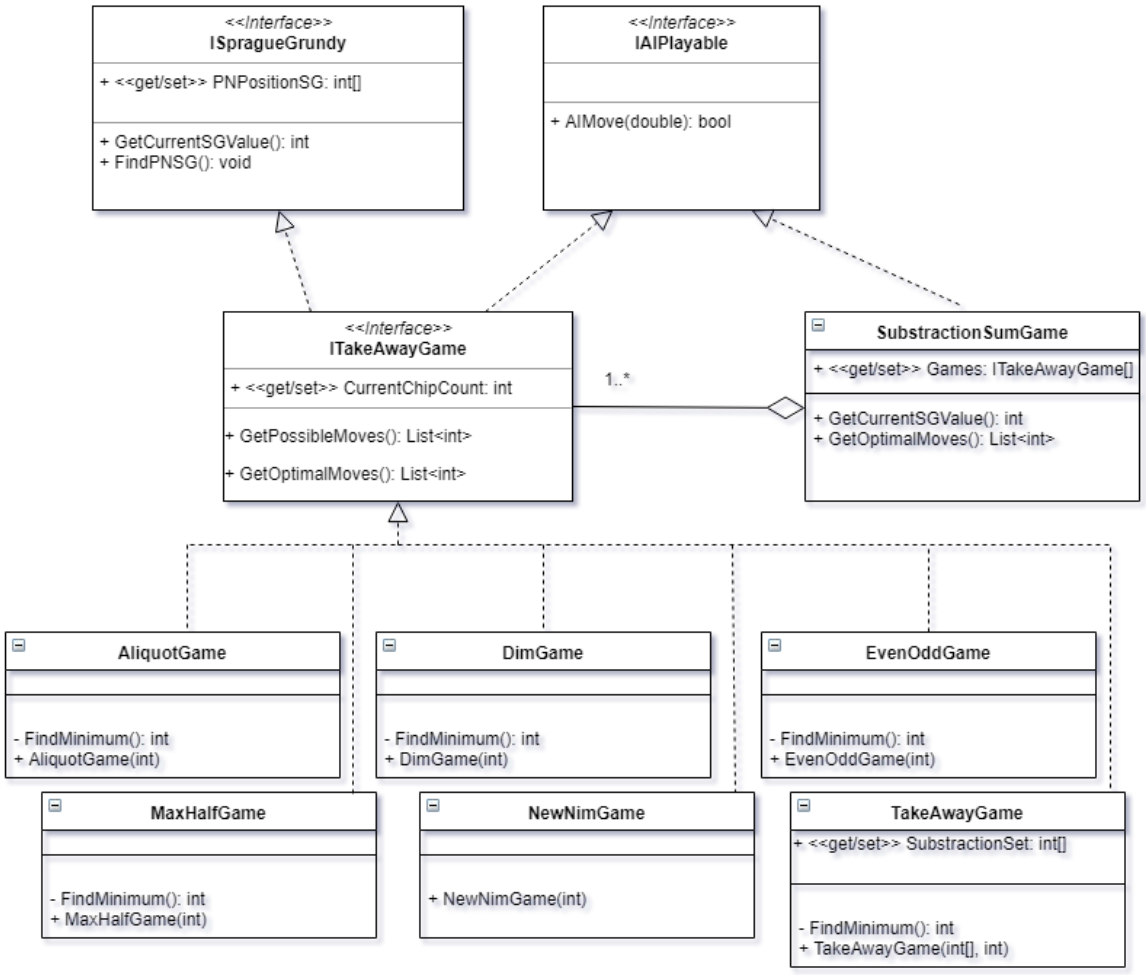
## 5.2 Statická struktura aplikace

Každý typ kombinatorické hry v aplikaci je reprezentován vlastní třídou. Následující diagram tříd (obrázek 4) popisuje aplikaci ze statického pohledu, poskytuje náhled na hierarchii těchto tříd, vztahy mezi nimi a také na jejich obsah.

1. Rozhraní *ISpragueGrundy* zajišťuje, aby třídy, které reprezentují jednotlivé hry, byly schopny využívat SG funkci pro výpočet SG hodnot a pozic.
2. *IAIPlayable* obsahuje funkci, díky které bude hru umět hrát také jednoduchá umělá inteligence.
3. *ITakeAwayGame* je abstrakcí pro herní prvky třídy, jako jsou proveditelné tahy, nebo také optimální tahy.
4. Třída *SubstractionSumGame* je reprezentací součtové hry, která se skládá z alespoň jedné triviální kombinatorické hry.

Zbylé třídy představují triviální kombinatorické hry, které mezi sebou nemají žádný vztah a dají se hrát samostatně. Dědí vlastnosti všech tří výše zmíněných rozhraní, avšak jejich implementace se kvůli různým pravidlům liší.

Existuje množství dalších nestranných odebíracích her s odlišnými pravidly. Při návrhu této aplikace je tedy důležité počítat s eventuální potřebou přidat do aplikace nové hry. Výše zmíněná rozhraní by podobná rozšíření aplikace měla usnadnit.




Obrázek 4: Diagram tříd

## 5.3 Interakce mezi objekty

V této kapitole se podíváme na to, jak se chovají objekty MVC architektury a jak mezi sebou spolupracují.

Obrázek 5 ilustruje, jak funguje architektura MVC v případě této aplikace. Tento konkrétní diagram zachycuje situaci, kdy si návštěvník stránky vytvoří hru a hraje ji proti druhému hráči na stejném počítači. Vzor MVC byl vybrán, protože díky rozdělení aplikace do tří hlavních komponent se značně zjednoduší procesy testování, debugovaní a údržby.

Komponenta *GameView* je zodpovědná za zobrazení uživatelského rozhraní klientovi. Využívá jazyk *Razor* [5], který umožňuje v jednom souboru současně využívat jazyk C# a HTML značky. Dokáže dynamicky vytvářet obsah webových stránek a je navíc možné sloučit kód ze strany serveru s obsahem pro klienta. Tato komponenta by měla obsahovat co nejméně logiky, nejlépe jen logiku pro uživatelské rozhraní. Většina logiky pro správné zobrazení grafického rozhraní uživateli je umístěna v souboru *site.js*. Tento soubor obsahuje několik JavaScriptových metod,

které lze díky funkcionalitě Razor stránek kombinovat nejen s prvky CSS a HTML, ale také s C# kódem.

*GameModel* zapouzdřuje většinu herní logiky. Obsahuje také některé informace o stavu aplikace, které pak mohou být předány do konkrétního GameView. Pro potřeby aplikace jsou nutné dva hlavní modely. Jeden pro reprezentaci triviálních odebíracích her a souvisejících dat a druhý pro reprezentaci součtových her.

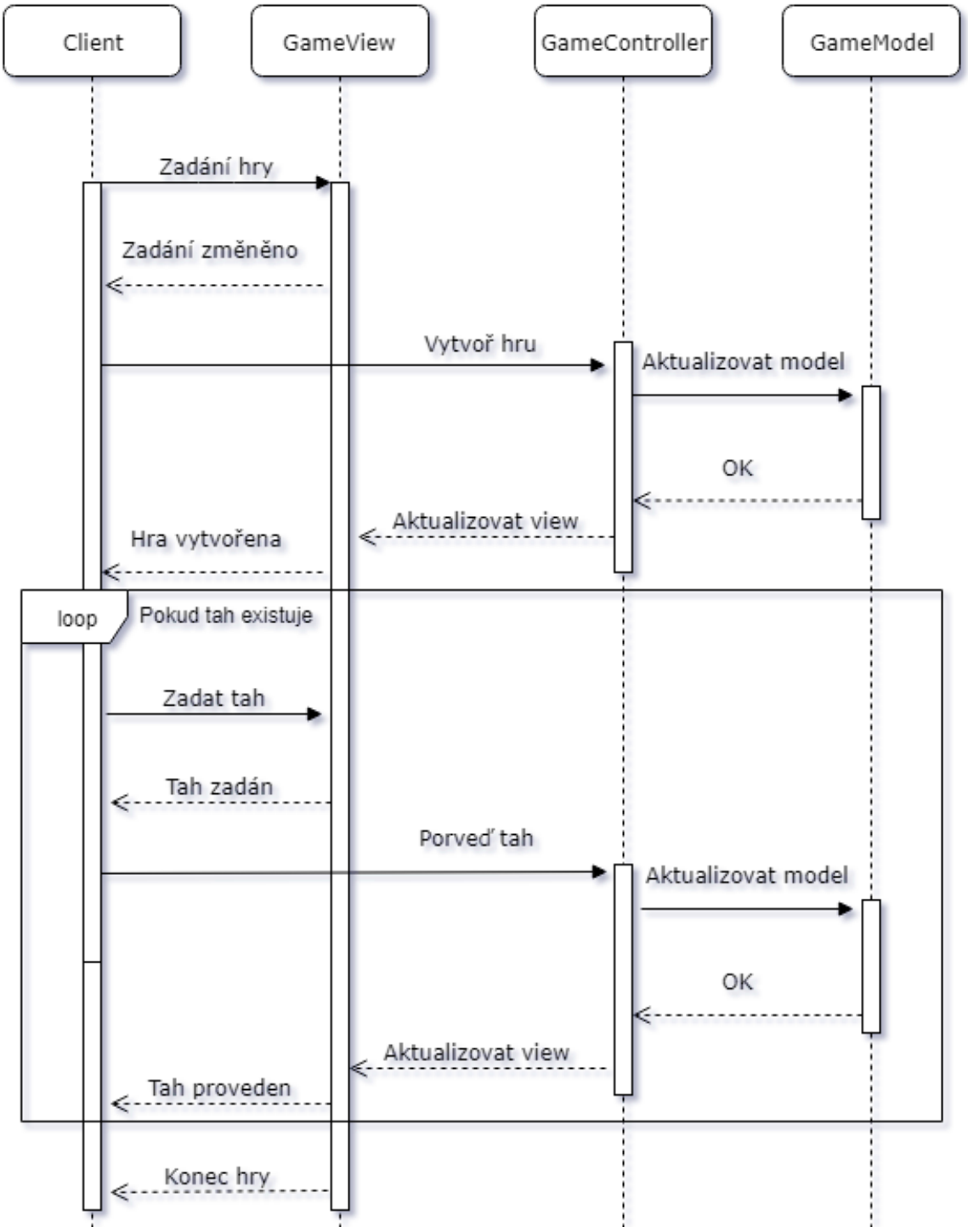
*GameController* se stará o interakci s klientem, říká komponentě GameModel, že má změnit svůj stav a také jak ho má změnit. Na rozdíl od GameView, kde jsou uživateli informace pouze zobrazovány, GameController tyto informace zpracovává, předává komponentě GameModel a následně posílá do konkrétního GameView, aby byly renderovány uživateli. Pro dobrou přehlednost a údržbu je vhodné implementovat několik controllerů, kde každý controller bude mít na starost funkčnost několika souvisejících stránek. V této aplikaci budou implementovány následující tři controllery.

1. **HomeController** - Bude pracovat se stránkami, které nepotřebují téměř žádnou logiku a pouze zobrazují statické informace. Jedná se například o úvodní stránku, nebo také o stránku s teorií.

2. **TakeAwayGamesController** - Bude spravovat stránky, které souvisí s konfigurací a hraním základních odebíracích her.

3. **SubstractionSumGamesController** - Tento controller manipuluje s tvorbou a hraním součtových her.

Všechny tyto controllery se skládají z metod, kde každá metoda je zodpovědná za nějaký úkon, jako je například automatické vytvoření hry, nebo provedení tahu umělé inteligence.




Obrázek 5: Sekvenční diagram MVC architektury aplikace

# 6 Implementace

V této kapitole se nachází samotná implementace webových stránek. Zde se podrobněji podíváme na vzhled a funkce uživatelského rozhraní, ale také na kód týkající se herní logiky a zobrazení herní strategie. Rozebereme také část implementace návrhového vzoru MVC a souvisejícího kódu v Razor stránkách.

## 6.1 Herní logika

Třídy na obrázku 4 reprezentují samotné kombinatorické hry a také velkou část herní logiky aplikace. Pro tento projekt bylo vybráno šest odlišných odebíracích her, je proto nutné implementovat šest tříd, které budou představovat korespondující odebírací hry. Pro pochopení kódu, který bude v této kapitole následovat si nejprve představíme pravidla pro odebírání žetonů v těchto šesti hrách. Pro všechny uvedené typy her platí, že pokud hráč nemůže provést legální tah, prohrál.

1. **Normální** - Na začátku se stanoví odebírací množina, neboli množina přirozených čísel, která reprezentují, kolik může hráč ve svém tahu odebrat žetonů.
2. **Nim** - Hráč může ve svém tahu odebrat libovolný počet žetonů (alespoň 1).
3. **Dim** - Hráč může odebrat $c$ žetonů z hromádky $n$ žetonů, pokud je $c$ dělitelem $n$, včetně 1 a $n$.
4. **Aliquot** - Hráč může odebrat $c$ žetonů z hromádky $n$ žetonů, pokud je $c$ dělitelem $n$, nemůže však odebrat celou hromádku.
5. **Sudá/lichá** - Hráč může odebrat sudý počet žetonů, pokud se nejedná o celou hromádku, nebo lichý počet žetonů, jestliže ji odebírá celou.
6. **Polovina** - Hráč může odebrat nanejvýš polovinu žetonů.

Všechny tyto hry je možné analyzovat pomocí funkce Sprague-Grundy, všechny jejich třídy tedy nepřímo dědí z rozhraní ISpragueGrundy. Jednou z metod tohoto rozhraní je *FindPNSG*. Následující ukázka kódu pochází z třídy *DimGame*.

```
public void FindPNSG()
{
   //Vytváří pole, kde každý prvek je jedna pozice
   PNPositionSG = new int[CurrentChipCount + 1];
   //Nultá pozice je vždy konečná, její SG hodnota je tedy 0
   PNPositionSG[0] = 0;
   //Získá SG hodnotu všech pozic
```

```
   for (int i = 1; i < PNPositionSG.Length; i++)
   {
        //Kolekce následovníků zkoumané pozice
        List<int> verteces = new List<int>();

        for (int j = 1; j <= i; j++)
        {
           if (i < j) continue;
           //Přidá do kolekce následovníky zkoumané pozice, podle pravidel Dim
           if (i % j == 0) verteces.Add(PNPositionSG[i - j]);
        }
        //Nalezení nejnižší hodnoty, která není mezi následovníky
        PNPositionSG[i] = FindMinimum(verteces);
   }
}.
```

Výpis 1: Metoda FindPNSG

V metodě FindPNSG je použita metoda *FindMinimum*. Jediný úkol této metody je navrátit nejmenší přirozené číslo, které není v kolekci, kterou přijímá jako argument. Je nutno podotknout, že kolekce musí být vzestupně seřazená, to ale díky vlastnostem metody FindPNSG splňuje.

```
public int FindMinimum(List<int> list)
{
   for (int i = 0; i <= list.Count; i++)
   {
      if (!list.Contains(i)) return i;
   }
   return -1;
}
```

Výpis 2: Metoda FindMinimum

Aby se mohli hráč, nebo umělá inteligence rozhodnout, jaký postup ve hře zvolí, musí nejdříve znát legální tahy. Pro tento problém třídy implementují metodu *GetPossibleMoves*. Následující kód je opět ze třídy DimGame.

```
public List<int> GetPossibleMoves()
{
   List<int> result = new List<int>();

    for(int i = 1; i <= CurrentChipCount; i++)
```

```
    {
      //Pokud číslo i beze zbytku dělí stávající počet žetonů, jedná se o
      //legální tah podle pravidel Dim
      if(CurrentChipCount % i == 0) result.Add(i);
    }
    return result;
}
```

Výpis 3: Metoda GetPossibleMoves

Součástí zadání je také požadavek, aby si hráč byl schopen zobrazit optimální strategii. Aby mu aplikace mohla takovouto strategii vytvořit, musí znát tahy, které vedou do vítězných pozic. Tuto vlastnost zajišťuje metoda *GetOptimalMoves*, která si nejprve pomocí metody GetPossibleMoves zjistí legální tahy a následně z nich vyfiltruje ty, které vedou do P-pozic.

```
public List<int> GetOptimalMoves()
{
    List<int> possibleMoves = GetPossibleMoves();
    List<int> optimalMoves = new List<int>();

    for (int i = 0; i < possibleMoves.Count; i++)
    {
      //Pokud provedeme legální tah (odebereme legální množství žetonů)
      //a dostaneme se jím do pozice s SG hodnotou 0, jedná se o optimální tah
        if (PNPositionSG[CurrentChipCount - possibleMoves.ElementAt(i)] == 0)
        {
            optimalMoves.Add(possibleMoves.ElementAt(i));
        }
    }
    return optimalMoves;
}
```

Výpis 4: Metoda GetOptimalMoves

Poslední metoda, kterou musí třídy implementovat, je *AIMove*. Jako argument přijímá pravděpodobnost pro optimální tah ve formě desetinného čísla. Pomocí náhodného generátoru čísel a tohoto atributu vyhodnotí, jaký tah má zvolit.

```
public bool AIMove(double probabilityForOptimal)
{
...
   List<int> optimalMoves = this.GetOptimalMoves();
   List<int> possibleMoves = this.GetPossibleMoves();
```

```
    if (possibleMoves.Count == 0) return false;

    int seed = DateTime.Now.Second;
    Random rnd = new Random(seed);
    double move = rnd.NextDouble();
    int randomMove = rnd.Next(possibleMoves.Count);

    if (move <= probabilityForOptimal)
    {
      //Zahraj první optimální tah. Pokud neexistuje, zahraj náhodný možný tah
      if(optimalMoves.Count == 0)
        {
         CurrentChipCount -= possibleMoves.ElementAt(randomMove);
        }
        else
        {
         CurrentChipCount -= optimalMoves.ElementAt(0);
        }
    }
    else
    {
      CurrentChipCount -= possibleMoves.ElementAt(randomMove);
    }
   return true;
...
}
```

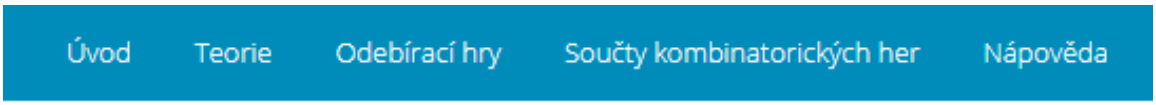
Výpis 5: Metoda AIMove

## 6.2 Uživatelské rozhraní

Webové stránky, které jsou předmětem této práce, jsou určeny jako učební pomůcka zejména pro studenty a pedagogy. Aby web opravdu usnadnil výuku a pomohl pochopit probíranou látku, je třeba dbát na jeho přehlednost a také intuitivní ovládání. Důraz je zároveň kladen na uzpůsobení uživatelského rozhraní pro mobilní telefony a jiná zařízení.

### 6.2.1 Navigační menu

Pro přístup k různým stránkám aplikace je určena navigační lišta v levém horním rohu každé stránky. Kód pro tuto komponentu je uložen v souboru *_Layout.cshtml*, na který odkazují ostatní soubory, které zajišťují zobrazení stránek. V případě změny pak stačí upravit kód pouze v jednom souboru. Menu můžeme vidět na obrázku 6.




Obrázek 6: Navigační lišta aplikace

Pokud si stránku otevřeme například na mobilním zařízení, menu se uzpůsobí velikosti obrazovky, viz. obrázek 7.


Obrázek 7: Navigační lišta aplikace u menších obrazovek

### 6.2.2 Úvod a teorie

Stránka *Úvod* má za úkol přivítat uživatele a nabídnout jednoduchý rozcestník, který odkazuje na různé části stránky se stručným popiskem, co na nich může najít.

Na stránkách *Teorie* se nachází informace, které jsou potřebné pro základní pochopení problematiky kombinatorických her. Předpokládá se, že po jejich nastudování bude student schopný vytvořit a hrát vlastní hru.

### 6.2.3 Odebírací hry

Na stránce *Odebírací hry* (obrázek 8) si uživatel může vytvořit vlastní odebírací hru. Na výběr má šest různých herních pravidel. Oponenta si může vybrat z několika obtížností umělé inteligence, nebo si zvolit hru proti druhému hráči. Výběr oponenta a pravidel je realizován přepínacími tlačítky. K dispozici jsou dva textboxy, první z nich určuje počet žetonů, druhý odebírací množinu hry.

Odebírací množinu může hráč nastavit pouze v případě, že si zvolí hru typu *Normální*. Uživatel může tlačítkem *Přidej* přidat do množiny prvek, tlačítkem *Reset* pak množinu vyprázdnit. V případě přidání prvku do množiny se prvek vypíše do textového pole umístěného o něco níže. Pomocí JavaScriptové funkce je zajištěno, že množina může konkrétní přirozené číslo obsahovat pouze jednou. Odebírací množiny zbylých her jsou automaticky definovány v jejich pravidlech, proto se při volbě jedné z těchto her korespondující textbox uzamkne. Uživatel má zároveň možnost vytvořit náhodnou hru, aby mohl ihned hrát.

Po vytvoření hry se uživateli zobrazí stránka se samotnou hrou. Na této stránce jsou dva panely. První panel poskytuje hráči základní informace, jako typ hry, počet žetonů, druh oponenta a pravidla hry. Druhý panel zobrazuje aktuální informace o hře, kolik žetonů zbývá a kdo je na tahu. Mezi těmito panely se nachází grafická reprezentace žetonů. Pod panely je výčet tahů, které může hráč aktuálně provést. Tahy jsou zobrazeny ve formě tlačítek. Tato část stránky je na obrázku 9.

Na obrázku 10 je tabulka, kterou si může uživatel rozkliknout, pokud si chce zobrazit informace o pozicích, strategii a optimálních tazích.

V okamžiku, kdy je hra u konce, se zobrazí jednoduchá stránka (obrázek 11), která vypíše vítěze a vybídne k vytvoření nové hry.

### 6.2.4 Součty kombinatorických her

Stránka *Součty kombinatorických her* dovoluje uživateli nakonfigurovat několik odebíracích her a následně z nich vytvořit součet kombinatorických her. Rozložení prvků se příliš neliší od stránky na obrázku 8. Rozdíl spočívá pouze v přidání dvou nových tlačítek. První z nich je tlačítko *Přidat hru*, které přidá právě nakonfigurovanou hru do seznamu her, které budou později sčítány. Druhé tlačítko je *Přidat náhodnou hru* a přidává náhodně nakonfigurovanou hru do téhož seznamu. Hry může uživatel odstranit kliknutím na tlačítko *Odebrat*, které se nachází u každé položky seznamu. Tlačítka a seznam můžeme vidět na obrázku 12.

Po vytvoření součtové hry se pro každou jednotlivou hru zobrazí panely a tlačítka, velmi podobně jako na obrázku 9. Jelikož si hráč může navolit sčítání většího množství her, přibyla v horním panelu rychlá navigace ve formě tlačítek, kdy každé tlačítko obsahuje název hry a počet zbylých žetonů (obrázek 13). Po kliknutí na tlačítko se stránka posune na vybranou hru.

## 6.3 Uchování stavu aplikace

Pro uchování stavu hry je použita vlastnost *Session* objektu *HttpContext*. Tato třída zajišťuje, že aplikace udrží určitá data v průběhu užívání aplikace. Klient dostane od aplikace *cookie* s jedinečným identifikátorem. Díky tomu pak aplikace s každým klientským požadavkem ví, odkud má data čerpat. Tyto cookies nejsou sdíleny napříč internetovými prohlížeči, proto prohlížeče nesdílí ani data. Lze nastavit, jak dlouho bude session aktivní, pokud nechceme využít výchozí nastavení na dvacet minut. Po skončení této doby se data automaticky mažou, pokud je chceme

smazat manuálně, musíme implementovat a následně zavolat *ISession.Clear*. Použití HttpContext.Session je nutné deklarovat v souboru *Startup.cs* pomocí metod *AddSession* a *UseSession* [12].

Následující úryvek kódu z controlleru *SubstractionSumGamesController* demonstruje uchování dat pomocí HttpContext.Session v aplikaci. Protože se při každém požadavku klienta na server vytvoří nový controller, vezme si nově vytvořený controller data právě odtud.

```
//Akce pro přidání hry do seznamu sčítaných her
public IActionResult AddGame(GameModel model)
{
...
   //Přiřazení dat z HttpConext.Session do objektu v controlleru
   List<GameModel> gameModels  = HttpContext.Session.
       GetObject<List<GameModel>>("GameModelList");

    if(gameModels == null)
    {
      gameModels = new List<GameModel>();
    }
...
   //Úprava objektu modelu
   gameModels.Add(model);
   //Uložení objektu do HttpContext.Session
   HttpContext.Session.SetObject("GameModelList", gameModels);

   return View(model);
}
```

Výpis 6: Kód pro uchování dat v rámci aplikace

## 6.4 Uživatelská příručka

Stránka *Nápověda* je určena pro nové uživatele. Jedná se o sadu ilustrací, které mají za cíl usnadnit uživateli orientaci v uživatelském rozhraní. Tato uživatelská příručka se rovněž nachází v příloze tohoto dokumentu.

## 6.5 Renderování stránky pomocí Razor view engine

V tomto projektu je každá stránka renderována pomocí tzv. *Razor view engine*. Díky tomu může bez problému kombinovat veškeré konstrukty programovacího jazyka C# se značkovacími jazyky

jako je například HTML. Zde si uvedeme několik úryvků kódu právě v syntaxi jazyka Razor. Prvním z úryvků je kód pohledu pro zobrazení odebírací hry.

```
//Model odeslaný controllerem, obsahuje data
@model GameModel
@{
   //Použití předdefinované šablony
    Layout = "~/Views/Shared/_Layout.cshtml";
}
...
   //HTML značky
   <b>Optimalni pocet zetonu k odebrani:</b>

   //Kód jazyka C\#
   @for (int i = 0; i < Model.OptimalMoves.Count; i++)
    {
       //Zobrazení optimálních tahů
       @Html.Raw(@Model.OptimalMoves.ElementAt(i) + ", ")
    }
...
```

Výpis 7: Kód pro zobrazení části stránky Odebírací hry

Razor má vlastní syntaxi také pro formuláře, textová pole, nebo rádiová tlačítka. Toho využívá například pohled, který má na starost zobrazení konfigurace odebírací hry. V následujícím úryvku kódu tohoto pohledu můžeme opět najít HTML značky, ale také JavaScriptovou funkci *disableSubstractionSet*, která v případě zvolení jiného, než normálního typu hry zablokuje textové pole pro úpravu odebírací množiny.

```
//Model odeslaný controllerem, obsahuje data
@model GameModel
...
<div class="container-fluid">
   //Začátek formuláře, který odešle vstupní informace do akce CreateGame
   //v controlleru TakeAwayGames
   @using (Html.BeginForm("CreateGame", "TakeAwayGames"))
   {
    ...
     <label class="btn btn-primary text-left ">
       //Rádiové tlačítko pro změnu typu hry
       @Html.RadioButtonFor(model => model.GameType, "normal",
        new {@type = "radio"
```

```
        ,@name = "options"
        ,@id = "normalRadio"
        ,onclick = "disableSubstractionSet(false)"
         }) Normální
      </label>
     ...
      //Tlačítko, které odešle formulář
     <button class="btn btn-lg btn-primary" type="submit">Vytvořit hru</button>
   }
</div>

//Tlačítko, které odkazuje na akci CreateRandomGame, tedy vytvoření náhodné hry
<div class="text-center btn-lg">
   <a class="btn btn-lg btn-primary"
   href="@Url.Action("CreateRandomGame")"
   role="button">Náhodná hra</a>
</div>
...
```

Výpis 8: Kód pro zobrazení části stránky Odebírací hry

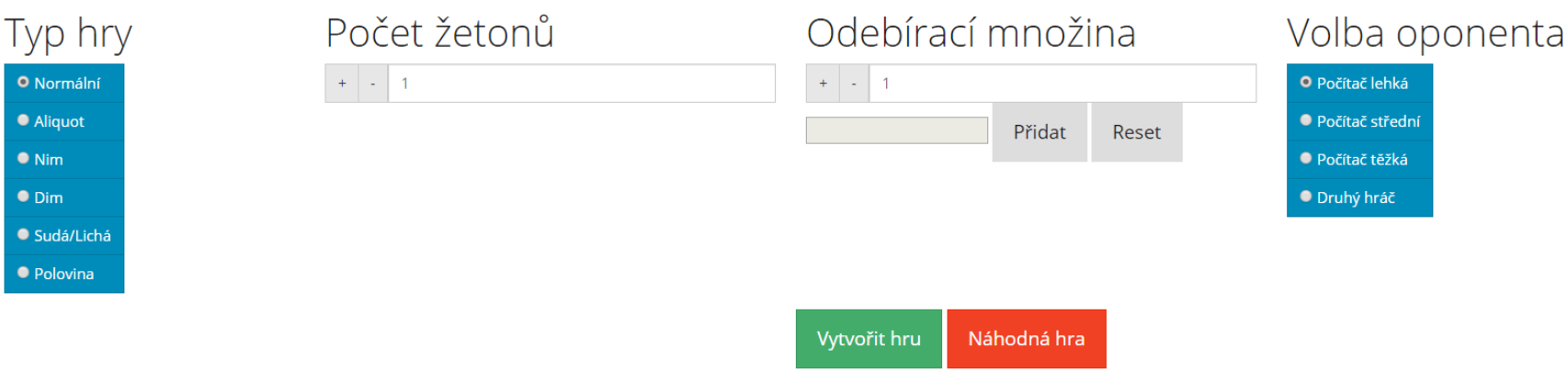
## 6.6 Nasazení aplikace na server

Webová aplikace, která je předmětem této práce, má být jednou z komponent na výukovém serveru pro teoretickou informatiku. Tato kapitola se ve stručnosti zabývá problematikou zprovoznění aplikace na webovém serveru. Základním požadavkem na server je zajištění podpory technologie ASP.NET Core.

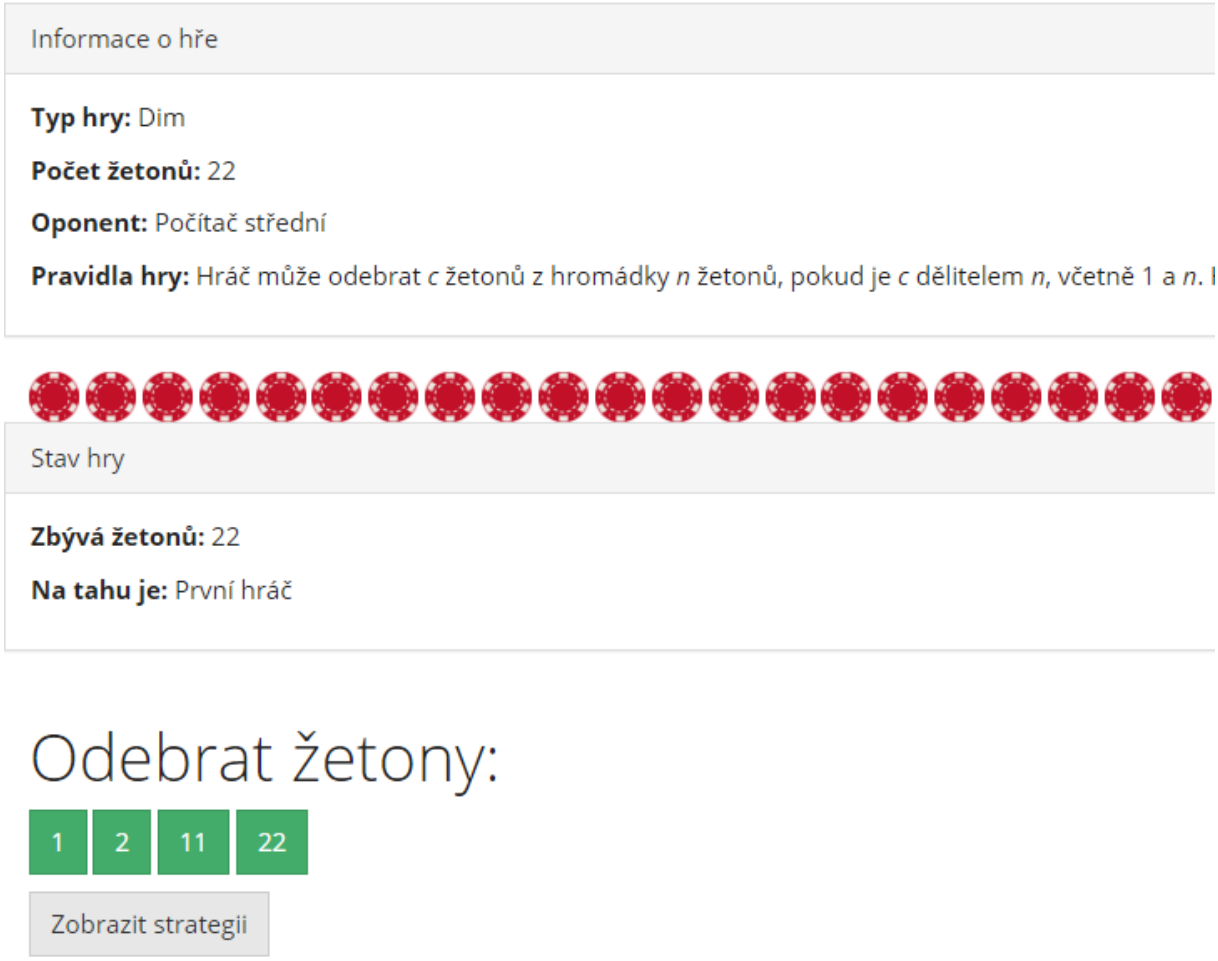
Pro účely nasazení aplikace byl vytvořen archiv, který lze rozbalit a nahrát do aplikačního adresáře na serveru, například pomocí FTP. Po nahrání souborů z archivu by měly stránky být plně funkční. Tento postup byl testován pomocí freehostingu ASPone [10]. Takto nasazená aplikace je dostupná na adrese `kombihry.aspfree.cz`. Archiv samotný je dostupný v elektronické příloze této práce, pod názvem *WebPackage.zip*.

## 6.7 XML dokumentace

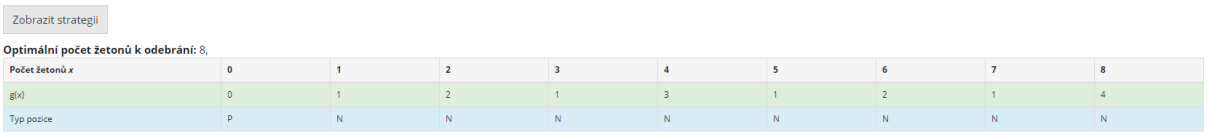
Pro dokumentaci zdrojového kódu byla použita vestavěná XML dokumentace Visual Studia. Soubor vygenerovaný touto službou se nachází v elektronické příloze pod názvem *CombinatorialGamesWebsite.xml*.



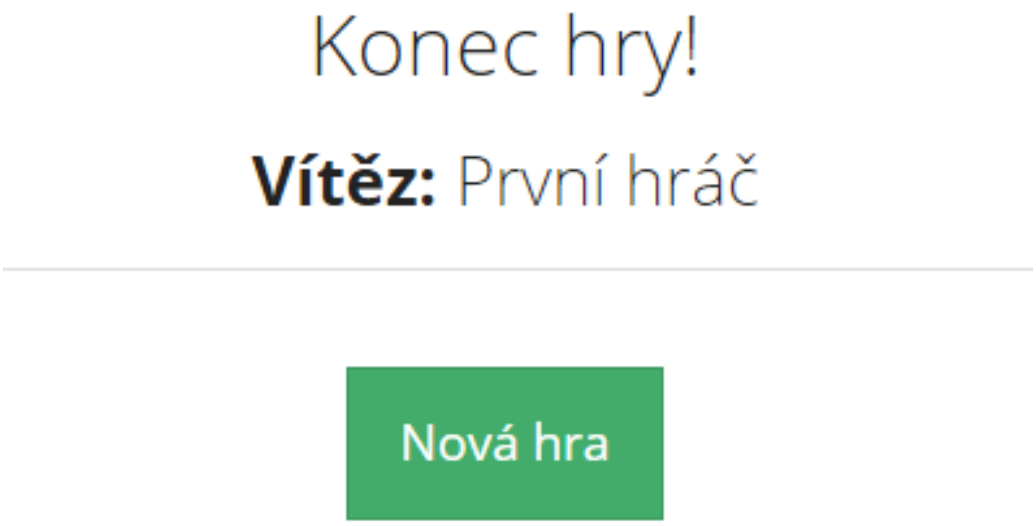

Obrázek 8: Rozložení stránky Odebírací hry



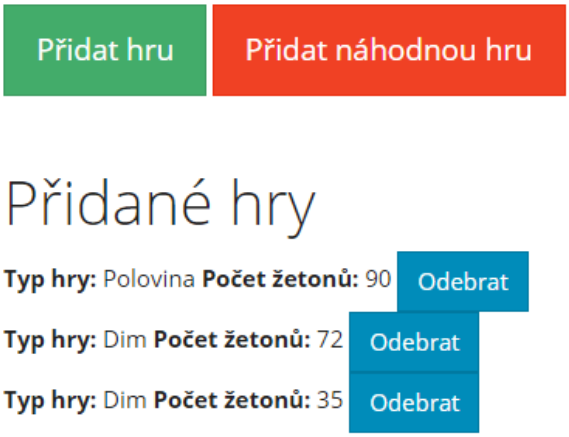

Obrázek 9: Panely s herními informacemi




Obrázek 10: Tabulka analyzující optimální strategii hry




Obrázek 11: Stránka, která se zobrazí po dohrání hry




Obrázek 12: Prvky pro přidání odebírací hry do hry součtové


Obrázek 13: Rychlá navigace pro přesun mezi hrami

# 7 Závěr

Náplní této práce bylo nastudovat a pochopit problematiku nestranných odebíracích her a následně vytvořit dynamické webové stránky, které by tento proces učení zájemcům o teoretickou informatiku urychlily a zpříjemnily. Stránky s totožným účelem už byly v rámci minulých let vytvořeny, mají však několik nevýhod, které byly v rámci nových stránek odstraněny.

Během vývoje aplikace jsem se blíže seznámil s řadou zejména webových technologií, především pak s frameworkem ASP.NET Core MVC, který je základním kamenem těchto stránek. Přesvědčil jsem se, že ačkoli tento projekt nepatří k těm největším, je důležité co nejdůkladněji uvážit návrh, ještě než se začne psát samotný kód.

Výsledná aplikace, podle mého názoru, splňuje všechny body zadání a může být nasazena na server s již existujícími projekty. Rozhodně však existuje prostor pro další vylepšení. Jako první se nabízí rozšíření aplikace o nové typy odebíracích her. Dalším, složitějším vylepšením, by mohlo být přidání možnosti hrát proti jiným hráčům přes internet. Nakonec by bylo vhodné vytvořit konfigurační soubor, ze kterého by stránky čerpaly nastavení, jako například pravděpodobnosti tahů umělé inteligence, nebo maximální počet sčítaných her. Aplikace umožňuje tyto změny provést, nejsou však nezbytně nutné, jedná se pouze o návrhy na budoucí vylepšení.

# Literatura


[1] FERGUSON, Thomas S. Game Theory [online]. [cit. 2019-03-26]. Dostupné z: https://www.math.ucla.edu/~tom/Game_Theory/comb.pdf

[2] BERLEKAMP, Elwyn R., John Horton CONWAY a Richard K. GUY. Winning ways for your mathematical plays. 2nd ed. Natick, Mass.: A.K. Peters, 2004. ISBN 1568811446.

[3] Bootstrap: The most popular HTML, CSS and JS library in the world. [online]. [cit. 2019-04-08]. Dostupné z: https://getbootstrap.com/

[4] Bootswatch: Free themes for Bootstrap [online]. [cit. 2019-04-10]. Dostupné z: https://bootswatch.com/

[5] Overview of ASP.NET Core MVC. Microsoft Docs [online]. [cit. 2019-04-10]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-2.2

[6] Introduction. Json.NET Documentation [online]. [cit. 2019-04-10]. Dostupné z: https://www.newtonsoft.com/json/help/html/Introduction.htm

[7] JQuery API. JQuery [online]. [cit. 2019-04-10]. Dostupné z: https://api.jquery.com/

[8] Visual Studio [online]. [cit. 2019-04-11]. Dostupné z: https://visualstudio.microsoft.com/cs/vs/

[9] BAŘÁK, Tomáš. Server pro podporu výuky teorie her [online]. Ostrava, 2013 [cit. 2019-04-20]. Dostupné z: http://hdl.handle.net/10084/98534. Diplomová práce. Vysoká škola báňská - Technická univerzita Ostrava.

[10] ASPone: Webhosting, freehosting, virtuální servery [online]. [cit. 2019-04-21]. Dostupné z: https://www.aspone.cz/cz/

[11] Draw.io: Online Diagramming [online]. [cit. 2019-04-21]. Dostupné z: https://about.draw.io/

[12] Microsoft Docs: Session and app state in ASP.NET Core [online]. [cit. 2019-04-25]. Dostupné z: https://docs.microsoft.com/cs-cz/aspnet/core/fundamentals/app-state?view=aspnetcore-2.2#session-state

# A Struktura elektronické přílohy

1. **Zdrojové kódy** - CombinatorialGamesWebsite
2. **Balíček pro nasazení na server** - WebPackage.zip
3. **Uživatelská příručka** - uzivatelska_prirucka.pdf
4. **Text práce** - 2019_zvo0017_BP_text.pdf
5. **XML dokumentace** - CombinatorialGamesWebsite.xml